To read the article online, visit http://www.4GuysFromRolla.com/articles/052009-1.aspx

Examining ASP.NET's Membership, Roles, and Profile - Part 14

By Scott Mitchell

Introduction

The ASP.NET Toolbox includes two Web controls for managing users' passwords: the ChangePassword control and the PasswordRecovery control. The ChangePassword control allows a user signed into the site to change their password by entering their existing password and their new, desired password. The PasswordRecovery control is used to reset or recover a user's password in the event that it has been forgotten. The PasswordRecovery control is used by anonymous users who need to be reminded of their password. Assuming that the Membership system is configured to require that users have a security question and answer (the default behavior), the user is presented with their security question and must correctly enter their security answer in order to have their password reset or recovered.

While there are two controls for managing passwords, there are no Web controls in the Toolbox for managing a user's security question and answer. In other words, there's no built-in control that allows a signed in user to change her security question and answer. The good news is that while no control offers this functionality it's not difficult to implement this feature ourselves. The `MembershipUser` class has a `ChangePasswordQuestionAndAnswer` method that modifies the security question and answer information using the configured Membership provider.

This article shows how to build a page that permits a signed in user to change their security question and answer, and a demo application is available for download at the end of the article that showcases this functionality in action. Read on to learn more!

Allowing a User to Change His Security Question and Answer

The `MembershipUser` class represents a user in the membership system. It has properties like `UserName`, `IsOnline`, `Approved`, `Email`, and `LastLoginDate`, and methods like `ChangePassword` and `ResetPassword`. One method in the `MembershipUser` class is `ChangePasswordQuestionAndAnswer`, which accepts three string input parameters - the user's password, the new security question, and the new security answer. The `ChangePasswordQuestionAndAnswer` then updates the security question and answer for the user account represented by the `MembershipUser` object and returns True if successful. If the password is incorrect, the `ChangePasswordQuestionAndAnswer` method returns False.

Creating a page where signed in users can change their security question and answer involves the following steps:

1. Build a user interface that prompts the user for their password, security question, and security answer
2. After the user supplies this information and clicks a Button, create a `MembershipUser` object for the currently logged in user and calls its `ChangePasswordQuestionAndAnswer` passing in the values entered by the user.
3. If the call to `ChangePasswordQuestionAndAnswer` succeeds, display a confirmation message; if it fails, alert the user that their password attempt was incorrect and let them try again.

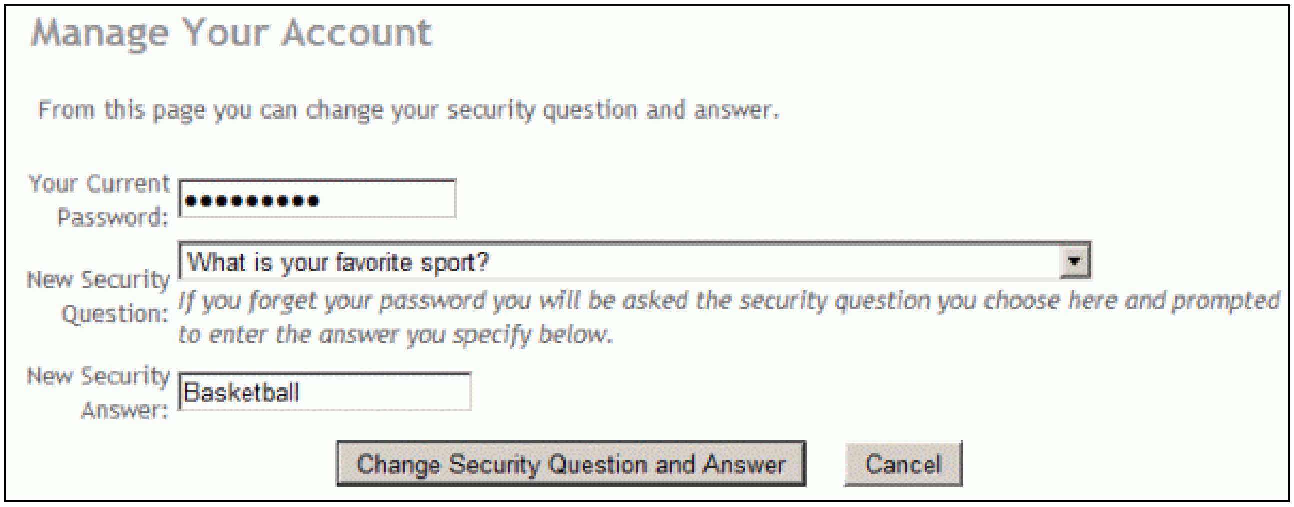
This article steps through each of these three steps. A demo application is available for download at the end of this article and builds on the work discussed in an earlier 4Guys article, Customizing ASP.NET's

CreateUserWizard Control To Display a Fixed Set of Security Questions.

Building the User Interface

The user interface that permits a user to change their security question and answer needs form field inputs for the user's password, security question, and security answer. If you allow free-form security questions, then all three inputs would be implemented using the TextBox control. However, if you limit the user to picking a security question from a pre-defined set of options (as was discussed in *Customizing ASP.NET's CreateUserWizard Control To Display a Fixed Set of Security Questions* and is the technique used in the demo application) then you would use TextBoxes for the password and security answer and a DropDownList control for the security question.

The following screen shot shows the user interface in action. In a nutshell, the UI is rendered in a five row `<table>`: one row for the password, one for the security question, another for the security answer, one for an error message (which is used if the supplied password is invalid), and the final row for the "Change Security Question and Answer" and Cancel Button controls. There are RequiredFieldValidator controls for the password and security answer TextBox controls and a ValidationSummary control that displays the validation error message (if any) in a client-side alert popup.


Manage Your Account

From this page you can change your security question and answer.

Your Current Password: ••••••••••

New Security Question: What is your favorite sport?
If you forget your password you will be asked the security question you choose here and prompted to enter the answer you specify below.

New Security Answer: Basketball

The above user interface is rendered inside a Panel control named `pnlChangeSecurityQA` that is, by default, visible. The page also contains another Panel control named `pnlSuccess` that contains a short message - "Your security question and answer was successfully changed" - and is hidden by default. The `pnlSuccess` Panel is shown (and the `pnlChangeSecurityQA` Panel hidden) after the user successfully changes their security question and answer.

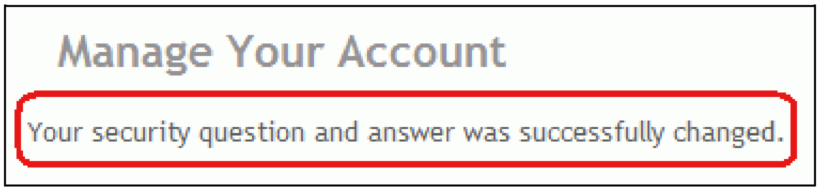
Keep in mind that this page is intended to be visited only by signed in users. Therefore, it should be placed in a folder that is only accessible to authenticated users (alternatively, you can modify `Web.config` to indicate that this specific file is only accessible to authenticated users). For more information on how to lock down folders and pages, refer to my tutorial on User-Based Authorization.

Calling the `ChangePasswordQuestionAndAnswer` Method and Displaying an Appropriate Status Message

When the user clicks the "Change Security Question and Answer" the page posts back and the Button's `Click` event handler fires. The event handler code is simple - we just need to get a `MembershipUser` object that represents the currently signed in user and then call the `ChangePasswordQuestionAndAnswer` method, passing in the values the user entered into the inputs. If the call succeeds, the `pnlSuccess` Panel is shown (and the `pnlChangeSecurityQA` Panel hidden). If it fails, a Label control on the page displays, "Your password was incorrect. Please try again."

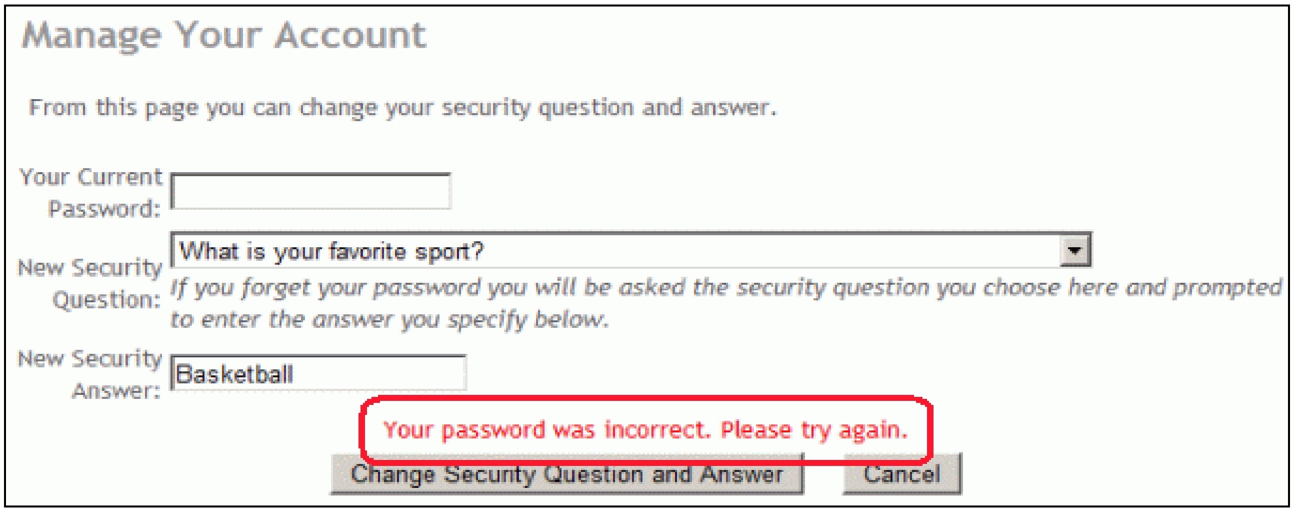
```
Protected Sub btnChangeSecurityQA_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnChangeSecurityQA.Click
   'Make sure page is valid
   If Not Page.IsValid Then Exit Sub

   'Change user's security question/answer
   Dim currentUserInfo As MembershipUser = Membership.GetUser()
   Dim success As Boolean = _
            currentUserInfo.ChangePasswordQuestionAndAnswer(Password.Text, _
                                                             Question.Text, _
                                                             Answer.Text)

   If success Then
      'Show the success Panel
      pnlSuccess.Visible = True
      pnlChangeSecurityQA.Visible = False
   Else
      'Invalid password
      ErrorMessage.Text = "Your password was incorrect. Please try again."
   End If
End Sub
```

Information about the currently signed in user is accessed via the call to `Membership.GetUser()`. Once this object is retrieves, its `ChangePasswordQuestionAndAnswer` method is invoked and passed the password, security question, and security answer entered by the user. If the supplied password was correct, the `ChangePasswordQuestionAndAnswer` method returns True and the `pnlSuccess` Panel replaces `pnlChangeSecurityQA`, displaying the message, "Your security question and answer was successfully changed."


Manage Your Account

Your security question and answer was successfully changed.

If the password is incorrect then the method returns False and the `ErrorMessage` Label control displays the message, "Your password was incorrect. Please try again."


Manage Your Account

From this page you can change your security question and answer.

Your Current Password:

New Security Question: What is your favorite sport?
If you forget your password you will be asked the security question you choose here and prompted to enter the answer you specify below.

New Security Answer: Basketball

Your password was incorrect. Please try again.

That's all there is to it! With that we now have a page that a signed in user can visit to change her security question and answer.

Having an Administrator Change a User's Security Question and Answer

As we just saw, in order to change his security question and answer the user must enter his password along with the new security question and answer. This password requirement is a security feature that protects users from having their security question and answer changed by another user when they step away from their computer (or when the use a public terminal and forget to sign out).

While this precaution seems reasonable, if you are tasked with creating a page that allows an administrative user the ability to change a user's security question and answer you'll quickly run into a showstopper - how is the administrator supposed to know the user's password? And without knowing the user's password the administrator cannot change the user's security question and answer! This challenge is similar to having an administrator change a user's password, because changing the password to a new one requires supplying the old one, as well.

I addressed the challenges and provided workarounds for having an administrator change a user's password in my Recovering and Changing Passwords tutorial. One of these workarounds can be applied to our problem, namely letting the administrator change a user's security question and answer. In short, you need to bypass the abstraction layer created by the Membership framework and work directly with the database (assuming you are using the SqlMembershipProvider provider). The SqlMembershipProvider database schema includes a stored procedure named `aspnet_Membership_ChangePasswordQuestionAndAnswer` that does not require that the user's password be supplied. Instead, you can supply just the username and the new security question and answer.

Keep in mind that bypassing the Membership API and going straight to the database is not encouraged. As I noted in the *Recovering and Changing Passwords* tutorial:

> By working directly with the database, the encapsulation provided by the Membership framework is shattered. This decision ties us to the SqlMembershipProvider, making our code less portable. Furthermore, this code may not work as expected in future versions of ASP.NET if the Membership schema changes. This approach is a workaround and, like most workarounds, is not an example of best practices.

Happy Programming!

- By Scott Mitchell

Further Reading
- Website Security Tutorials (VB & C# Versions Available)
- Customizing ASP.NET's CreateUserWizard Control To Display a Fixed Set of Security Questions
- User-Based Authorization Tutorial (VB Version) (C# Version)
- Recovering and Changing Passwords (VB Version) (C# Version)

Attachments
- Download the code used in this article

Article Information	
Article Title:	ASP.NET.Examining ASP.NET's Membership, Roles, and Profile - Part 14
Article Author:	Scott Mitchell
Published Date:	May 20, 2009
Article URL:	http://www.4GuysFromRolla.com/articles/052009-1.aspx